Task 2 Documentation

Mohammad Haris Ansari

October 2023

1 Introduction

The provided description outlines a binary comparison algorithm for comparing two integers a and b represented in binary form, each with a length of n bits. The key components of the algorithm are as follows:

Representation of Integers: The integers a and b are represented in binary form as sequences of bits, with n bits each. The bits are labeled from most significant (leftmost) to least significant (rightmost).

Output Variables:

- **FirstOutput (O_1):** This variable represents the result of a series of bitwise comparisons between the bits of a and b. It uses bitwise XOR ($\oplus$) and bitwise AND ($\cdot$) operations to compute a value of 1 if a is less than b, and 0 otherwise.
- **SecondOutput (O_2):** This variable represents whether the binary representations of a and b are identical.

Comparison Logic:

- If O_1 is equal to 1, it indicates that a is less than b.
- If O_1 is equal to 0 or O_2 is equal to 1, it indicates that a is equal to b.
- If neither of the above conditions is met, it indicates that a is greater than b.

Bitwise Comparison Loop: The algorithm appears to perform bitwise comparisons between the individual bits of a and b, starting from the most significant bit (MSB) and moving towards the least significant bit (LSB). The result of each bitwise comparison is accumulated in O_1.

Comparison Logic Details:

- For each bit position, the algorithm checks if a's bit is less than b's bit (e.g., $a_i < b_i$), and if so, it sets the corresponding bit in O_1 to 1.

- If the bits are equal (e.g., $a_i == b_i$), the algorithm checks the next bit positions (e.g., $(a_i \& a_{i+1}) == (b_i \& b_{i+1})$) and sets the corresponding bit in O_1 to 1 if the condition is met.
- This process continues for all bit positions from MSB to LSB.

Bitwise Comparison Completion: After completing the bitwise comparisons and setting the bits in O_1 accordingly, the algorithm also computes O_2 to check if a and b are entirely equal.

Comparison Result: The final result of the comparison is determined by the values of O_1 and O_2 based on the comparison logic described.

This algorithm provides a detailed and bitwise approach to comparing two binary representations of integers and correctly identifies whether one integer is less than, equal to, or greater than the other. It efficiently uses bitwise operations to avoid unnecessary comparisons when possible.

2 Python Code Explanation

the `find_negative_numbers(a)` function checks if there are any negative numbers in the input list `a`. It does so by iterating through the list and using the `find_the_largest_number` function to compare each element with zero. If it finds a negative number, it returns `True`. If it completes the loop without finding any negative numbers, it returns `False`.